Programación de Microservicios con Spring Boot y Red Hat SSO

Nivel Avanzado

Instructor: Carlos Carreño
ccarrenovi@gmail.com

Modulo 11 Seguridad avanzada

Objetivo: Introducción a Red Hat SSO y su integración con el desarrollo de aplicaciones con Spring Boot

Duración: 1h

Configuración de HTTPS en Spring Boot y Red Hat SSO

Configuración HTTPS en Spring Boot

✓Genera un certificado para Spring Boot

```
keytool -genkeypair -alias springboot -keyalg RSA -keysize 2048 \
  -storetype PKCS12 -keystore springboot-keystore.p12 \
  -validity 3650
```

continuación

- Configuración del **application.properties**

```
server.port=8443
server.ssl.key-store-type=PKCS12
server.ssl.key-store=classpath:springboot-keystore.p12
server.ssl.key-store-password=password
server.ssl.key-alias=springboot
```

Configuración HTTPS en Red Hat SSO / Keycloak

- Generar Keystore para RH-SSO
- Configuración en RH-SSO
- Agrega el socket binding
- Verifica

MFA y flujos de autenticación condicional

Red Hat SSO/Keycloak soporta MFA con:

- Google Authenticator
- FreeOTP
- OTP via Email (si habilitas el SPI de email)

1. Ingresa al admin **console** de Keycloak.
2. Ve a:

   Authentication → **Flows** → **Browser** → **Copy** (crear uno nuevo basado en Browser Flow).
3. Nombra este nuevo flujo:

   Browser with MFA
4. Modifica el flujo:

Step	Requirement
Cookie	ALTERNATIVE
Kerberos	DISABLED
Username Password	REQUIRED
OTP Form (One-Time Password)	REQUIRED

5. Asigna este flujo MFA a tu cliente o realm:

Realm Settings → Authentication → Browser Flow: Browser with MFA

Buenas prácticas para producción

- Keycloak / RH-SSO (Servidor de Identidad)

Recomendación	Descripción
HTTPS obligatorio	Configura HTTPS (TLS 1.2/1.3). No usar HTTP.
MFA activado para usuarios sensibles	Obliga OTP para roles como `admin`, `manager`.
Contraseñas robustas	Políticas estrictas: largo mínimo, complejidad.
Uso de flujos condicionales	MFA solo si: IP sospechosa / sin cookie previa / rol específico.
Revisar y auditar flujos personalizados	Verifica tus "Authentication Flows" activos.
Admin Console no expuesta a internet	Solo accesible desde red segura (VPN / Admin subnet).
Logs de eventos habilitados	Guarda eventos de login, error, MFA fallido, etc.

continuación

- Spring Boot (Resource Server / Backend)

Recomendación	Descripción
Validación JWT en servidor	Usar Spring Security OIDC Resource Server.
No confíes en headers Authorization manipulados	Delega siempre la validación a Spring Security.
Habilitar HTTPS (TLS)	Usa keystore real con certificado firmado CA.
CustomAuthenticationEntryPoint	Responde con JSON claro en errores JWT expirados.
Propagación segura de tokens (Feign, RestTemplate)	Incluye solo tokens válidos, nunca hardcodeados.

continuación

- Otras buenas prácticas

✓Usar claims custom para detección MFA en backend (acr, amr claims).

✓Manejo de Refresh Tokens solo en el cliente frontend (SPA, App móvil).

✓No guardar tokens en servidor backend.

✓Limitar duración de tokens (access_token_lifespan, refresh_token_lifespan) en Keycloak.

✓Manejar expiración de tokens con código personalizado en Spring (usando AuthenticationEntryPoint).